Overlapping Communication and Computation with High Level Communication Routines

Torsten Hoefler and Andrew Lumsdaine
Open Systems Lab
Indiana University
150 S Woodlawn Ave,
47405 Bloomington, IN, USA

Abstract

Collective operations and non-blocking point-to-point operations are two important parts of MPI that each provide important performance and programmability benefits. Although non-blocking collective operations are an obvious extension to MPI, there have been no comprehensive studies of this functionality. This dissertation will study non-blocking collective operations, integrating theory, practice, and application. We use a well-understood network model to found our theoretical analyses and we realize our communication operations as a portable library layered on MPI. A real-world quantum-mechanical application is used as a deployment and evaluation vehicle for our approach.

1 Introduction

The performance of parallel scientific applications is determined by per-CPU (sequential) performance and communication costs. While it is well understood how to optimize single-CPU performance, the task of minimizing communication overhead is still challenging. Although communication networks have evolved in the last years and are maturing quickly, the speed of light ultimately imposes a lower bound on communication latency. Thus, common optimizations to lower latency and improve bandwidth will soon find their natural limits—much like CPU frequency scaling which has recently stagnated. Increased parallelization can continue to provide performance improvements even though single processing elements are not getting faster. However, this only escalates the number of computing elements used to solve scientific problems. Since communication overhead grows with the number of messages and the number of messages often grows with the number of nodes, a steady decrease in efficiency seems to be unavoidable. Our research aims to overcome the effect of growing communication overhead due to those common limitations and growing processor numbers.

A possible solution to this problem is to hide communication costs by overlapping them with computation. Thus, we explore the benefits of non-blocking high-level communication operations. All implementations and tests are done as extensions to the widely-used and mature Message Passing Interface (MPI) standard [50]. The blocking collective operations currently defined by MPI offer a high-level interface to the user, insulating the user from implementation details and giving MPI implementers the freedom to optimize their implementations for specific architectures. That is, although collective algorithms do not provide unique functionality *per se* (they can be implemented manually with basic point-to-point operations), collective operations provide important advantages in programmability, safety (with regards to programming errors) and performance.

In this respect, collective operations can be compared to BLAS [47] operations. For example a high-level BLAS matrix multiply (e.g., DGEMM) operation could be easily composed of three nested loops[1], but the vendor supplied DGEMM implementation, because of special machine optimized tuning (e.g., cache/register optimization), usually provides much better performance. The same principle is used for collective operations as these operations can be optimized for the communication subsystem of a specific machine. Thus, many research groups have provided machine-optimized implementations and have investigated the optimal and non-trivial implementation of collective algorithms for particular machine architectures (cf. [29, 49, 54]).

The performance portability benefits of collective operations have long been recognized and collective operations play an important role in many applications. Consider, for example, a three-dimensional Fast Fourier Transformation implemented for a central-switch-based architecture (e.g., InfiniBand^{TM}). If the developer does not use the

[1]or lower level BLAS operations

MPI_Alltoall function, a fully connected send pattern (literally an all-to-all) should deliver the best performance[2]. However, if this implementation were ported to torus-based systems (e.g., an IBM BlueGene), the performance of the send-pattern mentioned above would be much worse than a torus-optimized MPI_Alltoall on that machine. However, because the collective operation interface is architecture independent, using it can avoid this performance decrease transparently, i.e., without changes to the user application.

A second MPI feature that plays a significant role in parallel programming is non-blocking point-to-point communication. These operations potentially allow communication and computation to be overlapped and thus they leverage hardware parallelism. The parallelism exists because most high-performance interconnect networks (like InfiniBand, Quadrics, Myrinet, Portals, or Ethernet with TOE) have their own communication co-processors that take the burden of message processing from the main CPU. However, this parallelism does not decrease the latency significantly, and it does not show its full potential if the programmer uses blocking send/receive. Non-blocking send/receive techniques allow the programmer to leverage the CPU during the asynchronous (and network-offloaded) message transmission. Several studies showed that the performance of parallel applications can be significantly enhanced with overlapping techniques (cf. [5]).

The work investigates the possibility of combining the advantages of collective operations with overlapped communication and computation in modern communication architectures. We propose a low-overhead and portable implementation of non-blocking collective operations that hides all the complexity of the internal implementation from the user. Furthermore, the research touches different neighboring areas such as network modeling, accurate measurement and performance prediction of collective operations, optimization of collective operations and general network optimization. The benefits of the new approach are shown with two real-world quantum mechanical applications, ABINIT and Octopus. Both programs solve the Schrödinger equation within the Density-Functional Theory (DFT). However, each program chooses a different approach for this solution and the application of non-blocking collectives differs fundamentally.

2 Related Work

The benefits of overlapping communication and computation have been leveraged by several researchers. Practical application performance has been shown to improve up to a factor of 1.9 [3, 48]. Dimitrov [11] explains the gains of overlapping on cluster systems while Kale et al. discusses the applicability of a non-blocking collective personalized communication for a set of applications in [44].

Further studies [42, 43, 46] analyze specific MPI implementations in detail and assess the possible benefits of overlapping on different systems. Some older studies (especially White et al. [43]) found that the investigated MPI implementations did not support asynchronous progress sufficiently. However, MPI implementations, as well interconnect networks, have evolved in the last years and support for overlap improved significantly [2, 46]. Brightwell et al. [6] analyze the source of different performance advantages and point out directions to further MPI optimization.

Several application studies have been conducted to analyze the possible benefits of overlapping for parallel applications. Sancho et al. [52] show a high potential of overlap for a set of scientific applications. Brightwell et al. [4] state clearly that many parallel applications could substantially benefit from non-blocking collective communication.

Automatic and semi-automatic transformations to parallel codes to enable overlapping of point-to-point communication have been proposed in many studies. However, none of them investigated transformations to non-blocking collective communication. Danalis et al. [10] even suggest replacing collective calls with non-blocking send-receive calls. This is clearly against the philosophy of MPI and destroys performance portability and many possibilities of optimization with special hardware support (cf. [28, 49]) completely.

Several parallel languages, like Split-C [9], UPC [41], HPF [12] or Fortran-D [13], have compilers available that are able to translate high-level language constructs into message passing code. Automatic schemes to enable overlapping within those schemes have also been proposed [10]. However, these compilers are only able to perform simple transformations and are not flexible enough for applications with non-trivial data-dependencies. For example, they are not able to handle a case where each process produces and communicates a different unpredictable amount of data in each iteration.

All approaches are either using overlapping techniques for point-to-point messages or optimize their codes with high-level communication routines. Some researchers even replaced high level operations by manual point-to-point implementation which is not portable among different machines. Our research however, investigates the non-trivial implementation and applicability of non-blocking collective operations and thus combines both non-trivial approaches.

3 Approach

A first step towards understanding the potential benefits of non-blocking collective operations is to understand the underlying communication networks in depth. However,

[2]in fact, many MPI implementations use this communication pattern to implement MPI_Alltoall on central-switch-based architectures

we do not want to limit our analysis to a particular network, thus, we choose an abstract network model to represent all needed properties of the networks. The choice of models is rather big, and the most common, the simple Hockney model [14] is not sufficient for our analyses because it only models the network transmission and not the CPU interaction. Kielmann's pLogP model [45] is rather complex and hard to use to model algorithms. Thus, we chose a modified Log(G)P model [8, 1] to represent all network parameters including the CPU overhead.

Most collective operations are implemented on top of hardware-specialized point-to-point algorithms. Those communication patterns can easily be modeled in LogGP terms and analyzed in theory. Such analyses for blocking collective operations have been done by Pjesivac-Grbovic [51] and our previous work [17]. Looking at the share of the CPU overhead and the LogGP parameters of modern communication networks reveals that the theoretical potential for overlapping in collective operations is very high. The possible benefit typically grows with the message size and first results show that modern networks can overlap more than 99% of the commmunication latency for reasonably-sized messages.

The next step is a prototypical implementation of collective operations that enables overlapping of communication and computation. This implementation should support all operations defined in the MPI standard and impose a very low CPU overhead.

The hardest step is to apply this principle to parallel applications. A first implementation should comprise several computational kernels to prove the benefits of overlapping. Real-world applications are to be targeted in a second step with the help of application developers. However, this task often requires changes to the optimized core algorithms in those applications and will thus be very time-consuming.

The adaptation of the reference implementation to a particular communication architecture will show several principles to optimize the performance (minimize the overhead) of non-blocking collective operations. This step might also require changes to the applications.

4 Research Results

Our research results over the last two years are presented in the following. The dissertation is expected to be finished in 12 to 18 months.

An important technological development, and one that has particular bearing on this work, is the observation that many high-performance networks do a significant amount of processing on the NICs, while the main CPU is only marginally involved. Our theoretical results begin with the LogP model family, which makes a clear distinction between time spent on the CPU (o) and time spent in the network (g,G,L). We chose the InfiniBand Architecture as an example network with which to implement our ideas. We investigated the prediction accuracy of the LogP model for small messages communicated with the Reliable Connection transport type of the InfiniBand network in [31]. This work was awarded the German PARS/GI "Junior Researcher Award 2005". Our findings showed the relative inaccuracy of the LogP predictions and we proposed a small modification to the original model to correct this error at the IPDPS 2006 conference in Rhodos, Greece [30]. Additional measurements, presented at the Parelec conference [38], showed that similar effects can be recorded with all InfiniBand transport types. We introduced a new portable, accurate, fast and congestion-free LogGP measurement method at the IPDPS conference 2007 [24] to investigate different networks. Our integration it in the open-source Netgauge network performance measurement tool which was released at the HPCC conference in Houston [27] 2007 allows other researchers to use it.

In the course of our research, we also investigated optimized collective communication for the InfiniBand network and with separate hardware support. A custom low-cost barrier for Open MPI was introduced at the ARCS 06 conference in Frankfurt [28, 37]. An overview and LogP modeling of several barrier implementations with practical validation in Open MPI is presented at the ICPP 05 in Oslo [17]. Novel algorithms that use the InfiniBand hardware specialties for implementation of MPI_Barrier and MPI_Bcast were introduced at the IPDPS conference 2006 [29] and 2007 [33] respectively. The work on MPI_Barrier [15] was awarded with the "Best Student Award 2005" of the Technical University of Chemnitz. We also stayed in close contact to the Open MPI team and proposed a novel idea for Open MPI's collective framework in [35].

After we explored different optimization techniques for blocking collectives and developed detailed network models which proved our theses, we took the first steps towards non-blocking collective operations. The benefits of non-blocking collective operations are discussed in [36] and a standard proposal, as extension to the MPI-2 standard was made [34]. We implemented LibNBC, a portable library that offers support for non-blocking collective operations [25]. The necessity of the standard extension is demonstrated at the EuroPVM/MPI User's group meeting 2007 [23] while the detailed performance of our implementation will be presented as a talk on the Supercomputing 2007 conference [26]. The idea of non-blocking collectives was presented at invited talks at the C&C Research Labs, NEC Europe, the Technical University of Dresden [16] and the HLRS Stuttgart. Our implementation achieves good overlap on MPI based systems. We also implemented an InfiniBand optimized version of LibNBC and developed a microbenchmark strategy [32] based on the latest findings in collective benchmarking. Microbenchmark results that compares the overhead of our MPI based implementation with the InfiniBand optimized version and the blocking MPI collectives on 64 InfiniBand nodes is shown in Figure 1.

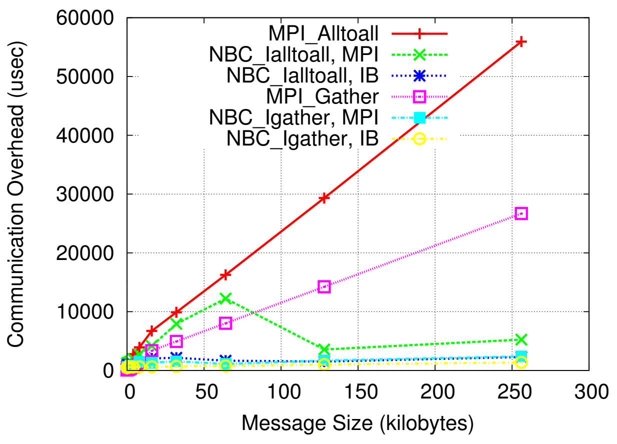

Figure 1. Gather and Alltoall overheads on 64 InfiniBand nodes

Those ideas were quickly picked up and adapted to different platforms by other researchers [53].

We proved the general usability and benefits of non-blocking collective operations in an article in the Elsevier Journal of Parallel Computing [19]. The results of this work show the benefits of non-blocking collective communication clearly. Figure 2 shows the speedup of a three dimensional Poisson solver and compares blocking to non-blocking communication over Gigabit Ethernet directly. The application ABINIT was analyzed for its parallel per-

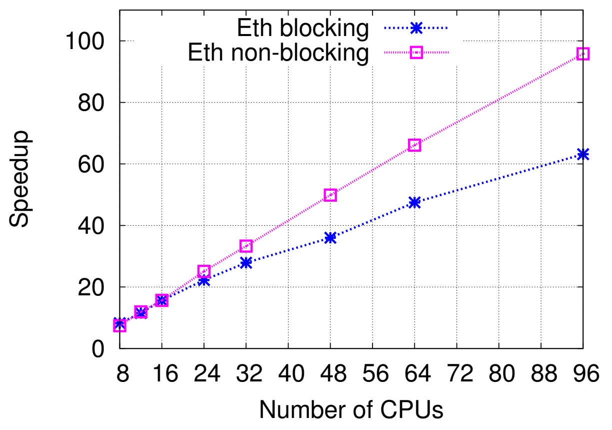

Figure 2. Parallel speedup of a 3d-Poisson solver, using blocking and non-blocking communication on Gigabit Ethernet

formance on different cluster systems in [21] and optimized during a research visit at the "CINECA Consorzio Interuniversitario" in Bologna, Italy [20] and [22]. Current ongoing work is to optimize three-dimensional Fast Fourier Transforms. First results that have been gathered during a research visit at the Guest Researcher at the "Commissariat a l'Energie Atomique (CEA)" in Paris, France can be found in [40]. Parallelization strategies for ABINIT were discussed in an invited talk at the " 3rd International ABINIT Developer Workshop" in Liege, Belgium [39].

5 Future Plans

Even though the research results are already used in real-world applications like Octopus [7], there are still many open questions. Our LogGP benchmark methodology is able to show the potential to overlap communication and computation. However, the current method is only accurate for blocking communication calls, To leverage overlap, one has to use non-blocking communications. While is is obvious that the parameters will not change significantly, they might vary slightly depending on the implementation and the underlying protocol. A more exact method to assess non-blocking communications is to be developed to predict communication performance accurately.

A detailed analysis of the new non-blocking collective operations in theory (network models) as well as in practice (applications) is planned for the near future. Using those results, LibNBC will be further optimized for the InfiniBand network and performance and overlap potential of different collective algorithms will be evaluated. To achieve maximum overlap, it is necessary to address the question of message progression which is relatively undefined in the MPI standard. We are planning to extend the InfiniBand implementation with an asynchronous progress thread that progresses outstanding messages.

Novel techniques to simplify the use of the new operations will also be investigated in detail. First steps in this direction are described in [18]. New non-blocking collective operations which are defined on processor grids (e.g., Cartesian MPI topologies) are developed together with application scientists to support novel architectures like BlueGene/L efficiently.

References

[1] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman. LogGP: Incorporating Long Messages into the LogP Model. *Journal of Parallel and Distributed Computing*, 44(1):71–79, 1995.

[2] C. Bell, D. Bonachea, Y. Cote, J. Duell, P. Hargrove, P. Husbands, C. Iancu, M. Welcome, and K. Yelick. An Evaluation of Current High-Performance Networks. In *IPDPS '03:*

Proceedings of the 17th International Symposium on Parallel and Distributed Processing, page 28.1, Washington, DC, USA, 2003. IEEE Computer Society.

[3] C. Bell, D. Bonachea, R. Nishtala, and K. Yelick. Optimizing Bandwidth Limited Problems Using One-Sided Communication and Overlap. In *Proceedings, 20th International Parallel and Distributed Processing Symposium IPDPS 2006 (CAC 06)*, April 2006.

[4] R. Brightwell, S. Goudy, A. Rodrigues, and K. Underwood. Implications of application usage characteristics for collective communication offload. *Internation Journal of High-Performance Computing and Networking*, 4(2), 2006.

[5] R. Brightwell, R. Riesen, and K. D. Underwood. Analyzing the impact of overlap, offload, and independent progress for message passing interface applications. *Int. J. High Perform. Comput. Appl.*, 19(2):103–117, 2005.

[6] R. Brightwell and K. D. Underwood. An analysis of the impact of MPI overlap and independent progress. In *ICS '04: Proceedings of the 18th annual international conference on Supercomputing*, pages 298–305, New York, NY, USA, 2004. ACM Press.

[7] A. Castro, H. Appel, M. Oliveira, C. A. Rozzi, X. Andrade, F. Lorenzen, M. A. L. Marques, E. K. U. Gross, and A. Rubio. Octopus: a tool for the application of time-dependent density functional theory. *PsiK Newsletter*, 73:145–173, 2006.

[8] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken. LogP: towards a realistic model of parallel computation. In *Principles Practice of Parallel Programming*, pages 1–12, 1993.

[9] D. E. Culler, A. C. Arpaci-Dusseau, S. C. Goldstein, A. Krishnamurthy, S. Lumetta, T. von Eicken, and K. A. Yelick. Parallel programming in Split-C. In *Supercomputing*, pages 262–273, 1993.

[10] A. Danalis, K.-Y. Kim, L. Pollock, and M. Swany. Transformations to parallel codes for communication-computation overlap. In *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 58, Washington, DC, USA, 2005. IEEE Computer Society.

[11] R. Dimitrov. *Overlapping of Communication and Computation and Early Binding: Fundamental Mechanisms for Improving Parallel Performance on Clusters of Workstations*. PhD thesis, Mississippi State University, 2001.

[12] High Performance Fortran Forum. *High Performance Fortran Language Specification, version 1.0*. Houston, Tex., 1993.

[13] S. Hiranandani, K. Kennedy, and C.-W. Tseng. Evaluation of compiler optimizations for fortran d on mimd distributed memory machines. In *ICS '92: Proceedings of the 6th international conference on Supercomputing*, pages 1–14, New York, NY, USA, 1992. ACM.

[14] R. Hockney. The communication challenge for MPP: Intel Paragon and Meiko CS-2. *Parallel Computing*, 20(3):389–398, March 1994.

[15] T. Hoefler. Evaluation of publicly available Barrier-Algorithms and Improvement of the Barrier-Operation for large-scale Cluster-Systems with special Attention on InfiniBand^{TM}Networks. Master's thesis, Chemnitz University of Technology, 2004.

[16] T. Hoefler. Non-blocking Collectives for MPI-2, 10 2007. Invited talk at the Dresden University of Technology, Center for Information Services and High Performance Computing (ZIH).

[17] T. Hoefler, L. Cerquetti, T. Mehlan, F. Mietke, and W. Rehm. A practical Approach to the Rating of Barrier Algorithms using the LogP Model and Open MPI. In *Proceedings of the 2005 International Conference on Parallel Processing Workshops (ICPP'05)*, pages 562–569, June 2005.

[18] T. Hoefler, P. Gottschling, and A. Lumsdaine. Transformations for enabling non-blocking collective communication in high-performance applications. Technical report, Open Systems Lab, Indiana University, 10 2007.

[19] T. Hoefler, P. Gottschling, A. Lumsdaine, and W. Rehm. Optimizing a Conjugate Gradient Solver with Non-Blocking Collective Operations. *Elsevier Journal of Parallel Computing (PARCO)*, 33(9):624–633, 9 2007.

[20] T. Hoefler, R. Janisch, and W. Rehm. Improving the parallel scaling of ABINIT. In *Science and Supercomputing in Europe - Report 2005*, pages 551–559. CINECA Conzorzio Interuniversitario, 12 2005.

[21] T. Hoefler, R. Janisch, and W. Rehm. A Performance Analysis of ABINIT on a Cluster System. In K. H. Hoffmann and A. Meyer, editors, *Parallel Algorithms and Cluster Computing*, pages 37–51. Springer, Lecture Notes in Computational Science and Engineering, 2006.

[22] T. Hoefler, R. Janisch, and W. Rehm. Parallel scaling of Teter's minimization for Ab Initio calculations. 11 2006. Presented at the workshop HPC Nano in conjunction with SC'06.

[23] T. Hoefler, P. Kambadur, R. L. Graham, G. Shipman, and A. Lumsdaine. A Case for Standard Non-Blocking Collective Operations. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface, EuroPVM/MPI 2007*, volume 4757, pages 125–134. Springer, 10 2007.

[24] T. Hoefler, A. Lichei, and W. Rehm. Low-Overhead LogGP Parameter Assessment for Modern Interconnection Networks. In *Proceedings of the 21st IEEE International Parallel & Distributed Processing Symposium*. IEEE Computer Society, 03 2007.

[25] T. Hoefler and A. Lumsdaine. Design, Implementation, and Usage of LibNBC. Technical report, Open Systems Lab, Indiana University, 08 2006.

[26] T. Hoefler, A. Lumsdaine, and W. Rehm. Implementation and Performance Analysis of Non-Blocking Collective Operations for MPI. In *In proceedings of the 2007 International Conference on High Performance Computing, Networking, Storage and Analysis, SC07*. IEEE Computer Society/ACM, 11 2007.

[27] T. Hoefler, T. Mehlan, A. Lumsdaine, and W. Rehm. Netgauge: A Network Performance Measurement Framework. In *High Performance Computing and Communications, Third International Conference, HPCC 2007, Houston, USA, September 26-28, 2007, Proceedings*, volume 4782, pages 659–671. Springer, 9 2007.

[28] T. Hoefler, T. Mehlan, F. Mietke, and W. Rehm. Adding Low-Cost Hardware Barrier Support to Small Commodity Clusters. In *19th International Conference on Architecture*

and Computing Systems - ARCS'06, pages 343–350, March 2006.

[29] T. Hoefler, T. Mehlan, F. Mietke, and W. Rehm. Fast Barrier Synchronization for InfiniBand. In *Proceedings, 20th International Parallel and Distributed Processing Symposium IPDPS 2006 (CAC 06)*, April 2006.

[30] T. Hoefler, T. Mehlan, F. Mietke, and W. Rehm. LogfP - A Model for small Messages in InfiniBand. In *Proceedings, 20th International Parallel and Distributed Processing Symposium IPDPS 2006 (PMEO-PDS 06)*, April 2006.

[31] T. Hoefler and W. Rehm. A Communication Model for Small Messages with InfiniBand. In *Proceedings PARS Workshop 2005 (PARS Mitteilungen)*, pages 32–41. PARS, 6 2005. (Awarded with the PARS Junior Researcher Prize).

[32] T. Hoefler, T. Schneider, and A. Lumsdaine. Accurately Measuring Collective Operations at Massive Scale. In *Accepted to the PMEO-PDS 08 workshop in conjunction with the 22nd International Parallel and Distributed Processing Symposium (IPDPS)*, April 2008.

[33] T. Hoefler, C. Siebert, and W. Rehm. A practically constant-time MPI Broadcast Algorithm for large-scale InfiniBand Clusters with Multicast. In *Proceedings of the 21st IEEE International Parallel & Distributed Processing Symposium*, page 232. IEEE Computer Society, 03 2007.

[34] T. Hoefler, J. Squyres, G. Bosilca, G. Fagg, A. Lumsdaine, and W. Rehm. Non-Blocking Collective Operations for MPI-2. Technical report, Open Systems Lab, Indiana University, 08 2006.

[35] T. Hoefler, J. Squyres, G. Fagg, G. Bosilca, W. Rehm, and A. Lumsdaine. A New Approach to MPI Collective Communication Implementations. In *Distributed and Parallel Systems - From Cluster to Grid Computing*, pages 45–54. Springer, 09 2006.

[36] T. Hoefler, J. Squyres, W. Rehm, and A. Lumsdaine. A Case for Non-Blocking Collective Operations. In *Frontiers of High Performance Computing and Networking - ISPA 2006 Workshops*, volume 4331/2006, pages 155–164. Springer Berlin / Heidelberg, 12 2006.

[37] T. Hoefler, J. M. Squyres, T. Mehlan, F. Mietke, and W. Rehm. Implementing a Hardware-based Barrier in Open MPI. In *Proceedings of 2005 KiCC Workshop, Chemnitzer Informatik Berichte*, November 2005.

[38] T. Hoefler, C. Viertel, T. Mehlan, F. Mietke, and W. Rehm. Assessing Single-Message and Multi-Node Communication Performance of InfiniBand. In *Proceedings of IEEE Inernational Conference on Parallel Computing in Electrical Engineering, PARELEC 2006*, pages 227–232. IEEE Computer Society, 9 2006.

[39] T. Hoefler and G. Zerah. Optimization of a parallel 3d-FFT with non-blocking collective operations, 01 2007. Invited talk at the 3rd International ABINIT Developer Workshop.

[40] T. Hoefler and G. Zerah. Transforming the high-performance 3d-FFT in ABINIT to enable the use of non-blocking collective operations. Technical report, Commissariat a l'Energie Atomique - Direction des applications militaires (CEA-DAM), 2 2007.

[41] P. Husbands, C. Iancu, and K. Yelick. A performance analysis of the berkeley upc compiler. In *ICS '03: Proceedings of the 17th annual international conference on Supercomputing*, pages 63–73, New York, NY, USA, 2003. ACM.

[42] C. Iancu, P. Husbands, and P. Hargrove. Hunting the overlap. In *PACT '05: Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques (PACT'05)*, pages 279–290, Washington, DC, USA, 2005. IEEE Computer Society.

[43] J. W. III and S. Bova. Where's the Overlap? - An Analysis of Popular MPI Implementations, 1999.

[44] L. V. Kale, S. Kumar, and K. Vardarajan. A Framework for Collective Personalized Communication. In *Proceedings of IPDPS'03*, Nice, France, April 2003.

[45] T. Kielmann, H. E. Bal, and K. Verstoep. Fast Measurement of LogP Parameters for Message Passing Platforms. In *IPDPS '00: Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing*, pages 1176–1183, London, UK, 2000. Springer-Verlag.

[46] W. Lawry, C. Wilson, A. B. Maccabe, and R. Brightwell. COMB: A Portable Benchmark Suite for Assessing MPI Overlap. In *2002 IEEE International Conference on Cluster Computing (CLUSTER 2002), 23-26 September 2002, Chicago, IL, USA*, pages 472–475. IEEE Computer Society, 2002.

[47] C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh. Basic Linear Algebra Subprograms for FORTRAN usage. In *In ACM Trans. Math. Soft., 5 (1979), pp. 308-323*, 1979.

[48] G. Liu and T. Abdelrahman. Computation-communication overlap on network-of-workstation multiprocessors. In *Proc. of the Int'l Conference on Parallel and Distributed Processing Techniques and Applications*, pages 1635–1642, July 1998.

[49] J. Liu, A. Mamidala, and D. Panda. Fast and Scalable MPI-Level Broadcast using InfiniBand's Hardware Multicast Support. Technical report, OSU-CISRC-10/03-TR57, 2003.

[50] Message Passing Interface Forum. MPI: A Message Passing Interface Standard. 1995.

[51] J. Pjesivac-Grbovic, T. Angskun, G. Bosilca, G. E. Fagg, E. Gabriel, and J. J. Dongarra. Performance Analysis of MPI Collective Operations. In *Proceedings of the 19th International Parallel and Distributed Processing Symposium, 4th International Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS 05)*, Denver, CO, April 2005.

[52] J. C. Sancho, K. J. Barker, D. J. Kerbyson, and K. Davis. MPI tools and performance studies—Quantifying the potential benefit of overlapping communication and computation in large-scale scientific applications. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 125, New York, NY, USA, 2006. ACM Press.

[53] J. C. Sancho, D. J. Kerbyson, and K. J. Barker. Efficient offloading of collective communications in large-scale systems. In *2007 IEEE International Conference on Cluster Computing (CLUSTER 2007), 17-20 September 2002, Austin, TX, USA*. IEEE Computer Society, 2007.

[54] S. S. Vadhiyar, G. E. Fagg, and J. Dongarra. Automatically tuned collective communications. In *Supercomputing '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM)*, page 3, Washington, DC, USA, 2000. IEEE Computer Society.